RECEIVED
CENTRAL FAX CENTER

SEP 15 2006

Yee & Associates, P.C.

4100 Alpha Road
Suite 1100
Dallas, Texas 75244

Main No. (972) 385-8777
Facsimile (972) 385-7766

Facsimile Cover Sheet

To: Commissioner for Patents for **Examiner Li B. Zhen** **Group Art Unit 2194**	Facsimile No.: **571/273-8300**
From: Carrie Parker Legal Assistant to Betty Formby	No. of Pages Including Cover Sheet: **34**

Message:

Enclosed herewith:

- Transmittal of Replacement Appeal Brief; and
- Replacement Appeal Brief.

Re: Application No. **10/085,547**
Attorney Docket No: **YOR920010667US1**

Date: Friday, September 15, 2006

Please contact us at (972) 385-8777 if you do not receive all pages indicated above or experience any difficulty in receiving this facsimile.	*This Facsimile is intended only for the use of the addressee and, if the addressee is a client or their agent, contains privileged and confidential information. If you are not the intended recipient of this facsimile, you have received this facsimile inadvertently and in error. Any review, dissemination, distribution, or copying is strictly prohibited. If you received this facsimile in error, please notify us by telephone and return the facsimile to us immediately.*

PLEASE CONFIRM RECEIPT OF THIS TRANSMISSION BY FAXING A CONFIRMATION TO 972-385-7766.

RECEIVED
CENTRAL FAX CENTER

SEP 15 2006

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of: **Bantz et al.**

Serial No.: **10/085,547**

Filed: **February 27, 2002**

For: **Automatic Provisioning for Subscription Computing**

54105
PATENT TRADEMARK OFFICE CUSTOMER NUMBER

Group Art Unit: **2194**

Examiner: **Zhen, Li B.**

Attorney Docket No.: **YOR920010667US1**

Certificate of Transmission Under 37 C.F.R. § 1.8(a)
I hereby certify this correspondence is being transmitted via facsimile to the Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450, facsimile number (571) 273-8300 on September 15, 2006.

By: _____
Carrie Parker

TRANSMITTAL OF REPLACEMENT APPEAL BRIEF

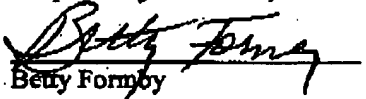
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:
ENCLOSED HEREWITH:

- Replacement Appeal Brief (37 C.F.R. 41.37)

No fees are believed to be necessary, as the fee for filing an Appeal Brief was previously paid in this application on November 2, 2005, with the filing of the first Appeal Brief. No additional fees are believed to be necessary. If, however, any additional fees are required, I authorize the Commissioner to charge these fees which may be required to Lenovo Deposit Account No. 50-3533. No extension of time is believed to be necessary. If, however, an extension of time is required, the extension is requested, and I authorize the Commissioner to charge any fees for this extension to Lenovo Deposit Account No. 50-3533.

Respectfully submitted,



Betty Formby
Registration No. 36,536
AGENT FOR APPLICANTS
Duke W. Yee
Registration No. 34,285
ATTORNEY FOR APPLICANTS
YEE & ASSOCIATES, P.C.
P.O. Box 802333
Dallas, Texas 75380
(972) 385-8777
ATTORNEY FOR APPLICANTS

RECEIVED
CENTRAL FAX CENTER

SEP 15 2006

Docket No. YOR920010667US1 ***PATENT***

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of: **Bantz et al.**	§	
	§	Group Art Unit: **2194**
Serial No. **10/085,547**	§	
	§	Examiner: **Zhen, Li B.**
Filed: **February 27, 2002**	§	
	§	
For: **Automatic Provisioning for Subscription Computing**	§	

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

54105
PATENT TRADEMARK OFFICE
CUSTOMER NUMBER

REPLACEMENT APPEAL BRIEF (37 C.F.R. 41.37)

Appellants request reinstatement of the Appeal. This brief is filed in response to the Final Office Action mailed June 15, 2006 and in furtherance of the Notice of Appeal, filed in this case on September 14, 2005.

No fees are believed to be necessary, as the fee for filing an Appeal Brief was previously paid in this application on November 2, 2005, with the filing of the first Appeal Brief. No additional fees are believed to be necessary. If, however, any additional fees are required, I authorize the Commissioner to charge these fees which may be required to Lenovo Deposit Account No. 50-3533. No extension of time is believed to be necessary. If, however, an extension of time is required, the extension is requested, and I authorize the Commissioner to charge any fees for this extension to Lenovo Deposit Account No. 50-3533.

REAL PARTY IN INTEREST

The real party in interest in this appeal is the following party: Lenovo (Singapore) Pte. Ltd., a corporation of Singapore, having a place of business at 9 Changi Business Park, Central 1, Singapore 486048.

RELATED APPEALS AND INTERFERENCES

With respect to other appeals or interferences that will directly affect, or be directly affected by, or have a bearing on the Board's decision in the pending appeal, there are no such appeals or interferences.

STATUS OF CLAIMS

A. TOTAL NUMBER OF CLAIMS IN APPLICATION

Claims in the application are: 1-50

B. STATUS OF ALL THE CLAIMS IN APPLICATION

1. Claims canceled: 3 and 48
2. Claims withdrawn from consideration but not canceled: None
3. Claims pending: 1-2, 4-47, and 49-50
4. Claims allowed: None
5. Claims rejected: 1-2, 4-47, and 49-50
6. Claims objected to: None

C. CLAIMS ON APPEAL

The claims on appeal are: 1-2, 4-47, and 49-50

STATUS OF AMENDMENTS

No amendments have been submitted since the final office action of June 15, 2006.

SUMMARY OF CLAIMED SUBJECT MATTER

A. CLAIM 1 - INDEPENDENT

The subject matter of claim 1 is directed to a computer-implemented method for creating customized disk images for loading software onto a computer. The method is shown in **Figures 5-7**, discussed on page 11, line 12 through page 13, line 17). The method contains the following steps:

- receiving software requirements for a given computer system from a plurality of users (**Figure 5**, step **200**, page 11, lines 14-16, **Figure 2A**, page 7, line 22 - page 8, line 15);
- determining (a) a plurality of existing software components that will fulfill the software requirements while addressing constraints and affinities between said plurality of software components and (b) a respective plurality of configuration options that reflect current best practices with regard to said plurality of software components (**Figure 5**, steps **203-204**, **Figure 6**, page 12, lines 6-24);
- generating a disk image containing said plurality of software components configured according to said respective plurality of configuration options (**Figure 7**, especially step **243**, page 12, line 25 - page 13, line 17.

B. CLAIM 11 - DEPENDENT

The subject matter of claim 11 is directed to the additional feature:

- generating a difference image that represents differences between the disk image and another existing disk image, whereby the another existing disk image may be updated to match the disk image by applying the difference image to the another existing disk image (page 15, lines 15-23).

C. CLAIM 16 - INDEPENDENT

The subject matter of claim 16 is directed to a computer-implemented method for creating a customized disk image for loading software onto a computer (**Figures 5-7**, page 11, line 12 - page 13, line 17). The method contains the following steps:

- parsing a plurality of inputs for a desired computer system to extract specifications regarding software (**Figure 5, steps 201 and 202, page 11, lines 17-22**).
- evaluating rules that apply to the plurality of inputs to derive a set of existing software components conforming to the specifications (**Figure 5, step 203, page 11, lines 23-24, Figure 3, 91, 92, and 93, page 10, lines 5-18**);
- evaluating additional rules that apply to the plurality of inputs to derive a set of configuration options conforming to the specifications (**Figure 5, step 203, page 11, lines 23-24, Figure 3, 91, 92, and 93, page 10, lines 5-18**);
- storing each software component from the set of software components on a storage device;
- configuring each software component stored on the storage device in accordance to the set of configuration options (**Figure 5, step 204, page 11, lines 23-24 and Figure 6, step 226, page 12, lines 19-20**);
- generating a disk image from contents of the storage device (**Figure 7, step 243; page 12, line 25 - page 13, line 13**).

D. CLAIM 19 - INDEPENDENT

The subject matter of claim 19 is directed to a computer program product. The computer program performs the method of claim 1 (page 16, lines 13-24).

E. CLAIM 34 - INDEPENDENT

The subject matter of claim 34 is directed to a data processing system that performs the method of claim 1 (**Figure 3, page 9, line 23 - page 10, line 23**).

FG. CLAIM 41 - DEPENDENT

The subject matter of claim 41 is directed to a further feature in the data processing system of claim 41:

- means for testing the disk image (page 14, line 19 - page 15, line 6).

G. CLAIM 44 - DEPENDENT

The subject matter of claim 44 is directed to a further feature in the data processing system of claim 34:

- means for generating a difference image that represents differences between the disk image and another existing disk image, whereby the another existing disk image may be updated to match the disk image by applying the difference image to the another existing disk image (page 15, lines 15-23).

GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

A. GROUND OF REJECTION 1 (Claims 1-2, 4-47, and 49-50)

Whether the examiner failed to state a prima facie obviousness rejection of claims 1-2, 4-47, and 49-50 as obvious over **Peterson** *et al.*, System Design Method, U.S. Patent No. 6,327,551, December 4, 2001 (hereinafter "**Peterson**") in view of **Fong** *et al.*, One-Click Deployment of Data Processing Systems, U.S. Patent Application Publication No. 2003/0055919, March 20, 2003 (hereinafter "**Fong**") under 35 U.S.C. § 103.

ARGUMENT

A. GROUND OF REJECTION 1 (Claims 1-2, 4-47, and 49-50)

A.1. Claims 1-2, 4-10, 12-28, 30-43, 45-47, and 49-50

Claim 1 is exemplary of this group of claims. Claim 1 is as follows:

> 1. A computer implemented method for creating customized disk images for loading software onto a computer, the method comprising the steps:
>
> receiving software requirements for a given computer system from a plurality of users;
>
> determining (a) a plurality of software components that currently exist and that will fulfill the software requirements while addressing constraints and affinities between said plurality of software components and (b) a respective plurality of configuration options that reflect current best practices with regard to said plurality of software components; and
>
> generating a disk image containing said plurality of software components configured according to said respective plurality of configuration options.

The Final Office Action, in rejecting this claim, states:

> 5. As to claim 1, Peterson teaches the invention substantially as claimed including a method for loading software onto a computer [col. 1, lines 5-8], the method comprising the steps:
>
> receiving software requirements [software requirements are documented in the form of a usage requirement specification; col. 1, line 63 - col. 2, line 12] for a given computer system [designing a product, system or service by deriving a requirement specification for said product, system or service from a user model; col. 3, lines 35 - 45] from a plurality of users [specification is an expression of the market opportunity in terms of the expected users goals, constraints imposed by users; col. 1, line 63 - col. 2, line 12];
>
> determining (a) a plurality of software components [components are functions and objects; col. 15, lines 29 - 41] that currently exist and [reusing components from the function layer (these components are functions and objects); col. 15, lines 29 - 42] that will fulfill the software requirements while addressing constraints and affinities between said plurality of software components [functional specification is produced from the requirements specification by means of a mapping from the requirements specification using or reusing components from the functional layer (these components are functions and objects); col. 15, lines 29 - 41] and (b) a respective plurality of configuration options that reflect current best practices with regard to said plurality of software components [determination of software requirements followed by

validation; col. 6, lines 1 - 17 and col. 6, line 58 - col. 7, line 16]. Peterson teaches a service packages [col. 13, lines 50 - 51] but does not teach generating a disk image containing said plurality software components configured according to said respective plurality of configuration options.

However, Fong teaches deployment of data processing systems with a specific set of software under the centralized control of a graphical user interface [p. 1-2, paragraph 0013] and generating a disk image [an automatic image capture of all hardware configurations and images from the selected reference data processing system; p. 8, paragraph 0055] containing said plurality software components configured according to said respective plurality of configuration options [user enters image capture information (e.g., name, description, and destination for the image) about data processing system; p. 8, paragraph 0055].

6. It would have been obvious to a person of ordinary skill in the art at the time the invention was made to apply the teaching of generating a disk image containing said plurality software components configured according to said respective plurality of configuration options as taught by Fong to the invention of Peterson because this allow an administrator to take a snapshot of an operating system configuration for a computer, including: base disk image, application packages, configuration settings, and specific hardware configurations [p. 1, paragraph 0008 of Fong]. This is driven from a central database containing unique parameters for each computer, including the rules that decide which images and software are applied to each computer [p. 1, paragraph 0008 of Fong].

Final Office Action mailed June 15, 2006.

The determination of "nonobviousness" is made after establishing the scope and content of prior art, the differences between the prior art and the claims at issue, and the level of ordinary skill in the pertinent art. *Graham v. John Deere*, 383 U.S. 1 (1966). In addition, all limitations of the claimed invention must be considered when determining patentability. *In re Lowry*, 32 F.3d 1579, 1582, 21 U.S.P.Q.2d 1031, 1034 (Fed Cir. 1994). A *prima facie* case of obviousness is established when the teachings of the prior art itself suggest the claimed subject matter to a person of ordinary skill in the art. *In re Bell*, 991 F.2d 781, 783, 26 U.S.P.Q.2d 1529, 1531 (Fed. Cir. 1993). The mere fact that references can be combined or modified does not render the resultant combination obvious unless the prior art also suggests the desirability of the combination. *In re Mills*, 916 F.2d 680, 16 USPQ2d 1430 (Fed. Cir. 1990).

The examiner has failed to state a *prima facie* obviousness rejection in view of **Peterson** and **Fong** at least because (a) specific features in the claims are not met, and (b) one of ordinary skill in the art would not combine these two references when they are looked at as a whole.

A.1. (a) Features Not Met

Peterson does not disclose or suggest "*determining ... a plurality of software components that currently exist and that will fulfill the software requirements*", as claimed. **Peterson** also does not disclose or suggest that the determining is performed "*while addressing constraints and affinities between said plurality of software components*", as claimed. **Peterson** further does not disclose or suggest "*determining ... a respective plurality of configuration options that reflect current best practices with regard to said plurality of software components,*" as claimed. Furthermore, **Fong** does not cure the lack of disclosure in **Peterson** in this regard. Therefore, the proposed combination of references when considered as a whole does not teach or suggest all of the features of the claims. Accordingly, the examiner has failed to state a *prima facie* obviousness rejection of these claims. Applicants address each of these features in turn.

Regarding the first of these features, the invention as recited in claim 1 identifies software components that meet at least two criteria – the software components *currently exist* and *will fulfill the software requirements* that have been received from the users. The occurrence of these two criteria is a common scenario in creating both business and personal computer systems, where a need exists for cost-effective methods to determine a best solution from the many choices facing the consumer. The above features of claim 1 directly address this need.

In contrast, **Peterson** teaches the design of new telecommunications capabilities and determines software requirements that may well need to be written. **Peterson** notes:

> The invention relates to a method of designing, systems, products and services, particularly information systems used in the design of new telecommunications services, and a design engine for implementing the method.

Peterson, column 1, lines 5-8, emphasis added.

> Referring to the drawings, and FIG. 1 in particular, there is shown a diagrammatic representation of the waterfall software development life cycle model. It is important to understand this technique to appreciate the relationship of the invention to the prior art and the advantages of the present invention over the prior art.

Peterson, column 5, lines 60-62, emphasis added.

> The design method of the present invention is an analytic technique which builds on and improves the waterfall approach and uses prototyping.

Peterson, column 1, line 63-65.

Peterson is directed to a method for assisting new designs, especially in the telecommunications industry. After Appellants proved these facts in the previous response, the Examiner replied:

> ... examiner respectfully disagrees and ... notes that Peterson teaches the software requirements creates the function specification which identifies components from the functional layer that can be reused [col. 15, lines 29-41]. The components from the functional layer correspond to the claimed software components because the functional layer contains the functions and objects of the chosen implementation technology [col. 16, lines 8-23]. Since components from the functional layers are reused, the components already exist. Therefore, Peterson determines a plurality of software components that currently exists and that will fulfill the software requirements.

Final Office Action mailed June 15, 2006, page 2, item 2.

Although **Peterson** discloses that some components can be reused, this fact does not mean that this reference can find existing components that also fulfill the software requirements. In fact, **Peterson** discloses that this is not true, noting:

> At this stage of the design it becomes important to discriminate between service creation and design of systems and products. If a service is being creating, the service states correspond to service features and the service tasks correspond, for example, to changing the state of a service component and can be regarded as equivalent to utilising a feature manager. However, if the service in question cannot be properly decomposed into service components corresponding to service states the design has to be carried further into the system or even into product design. This of course is strongly related to the subgoal decomposition tree. <u>If a breakdown of a market opportunity main goal into a set of available services is available it is only necessary to design the service manager. If the breakdown results in a set of existing services or features together with some services not currently available, then it will be necessary to implement these services in detail.</u>

Peterson, column 13, line 55 through column 14, line 4.

Peterson is providing new services and systems and discloses that even if a desired service can be met by combining existing services, there is still the need to <u>write</u> a manager to

oversee the combination. Since this reference discloses that even the creation of new capabilities by combining existing services requires at least one new component, **Peterson** does not disclose or suggest the claimed feature that the *software requirements* can be met by *software components that currently exist*. **Fong** is also devoid of disclosure in this regard, as discussed below.

Regarding the second feature discussed, **Peterson** does not disclose that the selection of software must also address "*constraints and affinities between said plurality of software components*". Against this feature, the rejection cites the following:

> Each category has a corresponding layer. These layers may contain reusable information, components, supply structures and entities. In this way a conceptual specification can be produced from the requirements analysis by using or reusing concepts from the conceptual layer. Similarly the functional specification is produced from the requirements specification by means of a mapping from the requirements specification using or reusing components
>
> from the functional layer (these components are functions and objects). This process also holds good for the interface and device layers. Of course, when no reusable component exists design must be performed at the lower levels.

Peterson, column 15, lines 29-41.

This excerpt addresses requirements, but neither discloses nor suggests that the software components necessarily interact with each other to varying degrees and that any software component may affect the ability of another component to execute properly. **Peterson** does not address the issues of *constraints* and *affinities* that can exist when deciding the components of the system. Thus, **Peterson** does not disclose the claimed feature of "*addressing constraints and affinities between said plurality of software components*". **Fong** is also devoid of disclosure in this regard, as discussed below.

Finally, **Peterson** does not address *determining ... a respective plurality of configuration options*, as recited in claim 1. The rejection reads this step on **Peterson**, citing *[determination of software requirements followed by validation; col. 6, lines 1 - 17 and col. 6, line 58 - col. 7, line 16]* as showing this determining step. **Peterson** states:

> Feedback between adjacent steps is possible via paths 15-20. Each step includes a validation or testing process which may reveal errors or the need for design modification. The individual steps in the design process are:
>
> system feasibility study, followed by a validation process, 1;

> determination of software requirements followed by validation, 2;
> preliminary design followed by validation, 3;
> detailed design followed by validation, 4;
> coding and debugging phase followed by a development test, 5;
> system test and pre-operation phase followed by validation, 6; and
> finally operation and maintenance phases followed by
> revalidation, 7.

Peterson, column 6, lines 1-17.

> After the market opportunity has been described, task analysis is used to breakdown or decompose the main goal into a set of sub-goals or sub-sub-goals. A functional requirement analysis of the goal tasks then identifies the service states, i.e the states of the work system that enable users to complete their goals, the associated service tasks, i.e. the set of tasks necessary to produce the service states, and the requirements placed on these tasks. Finally, what may be described as a top down interface analysis is applied, in which the service states and their associated service tasks are further decomposed into interface objects and interface procedures using a layered user interaction reference model or UIRM. From this a number of specifications or representations of varying degrees of abstraction and detail together with their corresponding prototypes, validation, verification and usability evaluation can be produced. Of course the final system or a specification thereof can also be produced.

Peterson, column 6, line 5 through column 7, line 7.

In the cited text **Peterson** is validating steps in the design process. The validation of software requirements is not the same as determining configuration options. **Peterson** discloses that validation is used to "reveal errors or the need for design modification" (**Peterson**, column 6, lines 2-4). This teaching is in contrast to configuration options, which are used to optimize or customize programs for a user or situation. Thus, **Peterson** does not address *determining ... a respective plurality of configuration options*, as claimed. **Fong** is also devoid of disclosure in this regard, as discussed below.

Although **Peterson** is cited to show these steps, **Fong** also does not show these steps. When **Fong** captures the images that are to be written to disk, this process is user-directed. **Fong** states that:

> [0055] FIG. 7 illustrates a flow chart of image capture, in accordance with one preferred embodiment of the invention. The method starts in operation 702. In operation 704, the user selects image capture (e.g., from a GUI or menu). Operation 706 is next, where the user selects the reference data processing system. Operation 708 is next, where the user enters image capture information (e.g., name, description, and

> destination for the image) about data processing system(s). Operation 710 is next, where a test determines if a default image capture, or a customized image capture, is to be made. If the test of operation 710 determines it is a default image capture (i.e., if the user selected the default image capture option), then operation 712 is next, where there is an automatic image capture of all hardware configurations and images from the selected reference data processing system. If the test of operation 710 determines it is a customized image capture, then operation 714 is next, where the user selects the customize option. Operation 716 is next, where hardware, base software image, or incremental image capture options are selected. Operation 720 is next, where the image capture status is displayed. Operation 722 is next, where the final report on the image capture is displayed. Operation 724 is next, where the method ends.

Fong, paragraph 0055, emphasis added.

Fong expects the user to select the software that will be written to disk, as shown above. **Fong** does not determine "*software components*", or address "*constraints and affinities*", or determine "*configuration options*" using a computer implemented method, as claimed. Because neither **Peterson** nor **Fong** performs these recited actions, the claimed features are not met by the references relied on. For this reason, the proposed combination when considered together as a whole does not teach or suggest all of the features of claim 1. Therefore, the rejection with respect to claim 1 and the remaining claims in this grouping of claims should be overturned.

A.1. (b) Lack of a motivation when the combination is considered as a whole

In addition to the features of the invention that are not shown by **Peterson** and **Fong**, there are considerable differences between **Peterson** and **Fong**. For this reason, one of ordinary skill in the art would not seek to combine these references when they are considered as a whole.

In considering the references as a whole, one of ordinary skill in the art would look at the problems recognized and solved. **Peterson** is directed towards assisting the design of new telecommunications services, noting, "*In today's markets it is frequently necessary to produce extremely complex custom-tailored systems with great expedition. This requires the use of efficient methods for the design of systems and products*" (**Peterson**, column 2, lines 11-14). In contrast, **Fong** is directed towards the problems in the deployment of data processing systems, noting a "*major problem inhibiting deployment of a group of data processing systems is the complexity of setting up the software and parameters of a larger group of data processing*

systems" (**Fong**, paragraph 0012). These two problems are unrelated. For this reason, one of ordinary skill in the art would not be motivated to combine these two references when they are read a whole. Instead, one of ordinary skill in the art would look to references that discuss similar problems to the one being solved.

As further support, the two cited references provide entirely different solutions. **Peterson** provides "*an analytic technique which builds on and improves the waterfall approach and uses prototyping*" (**Peterson**, column 1, lines 63-65) and further notes that the "*present invention has a number of novel aspects which contribute to its value and which are not employed in other design processes*" (**Peterson**, column 2, lines 27-29, emphasis added). In contrast, **Fong** provides "*a method ... to facilitate the intelligent deployment of one or more data processing systems*" (**Fong**, Abstract) and notes that the "*present invention provides a comprehensive method and system to facilitate the intelligent deployment of a group of data processing systems with a specific set of software, hardware firmware versions, and parameters under the centralized control of a graphical user interface*" (**Fong**, paragraph [0013]). Thus, **Peterson** is directed to a design solution, while **Fong** is directed to the deployment of software onto computers. Because these problems are wholly distinct, no motivation exists for one of ordinary skill to combine the references in the manner suggested by the examiner. Accordingly, the examiner has failed to state a prima facie obviousness rejection against claim 1 and the remaining claims in this grouping of claims.

The Examiner responded to the above arguments, replying:

> As to argument (2) [motivation to combine], examiner respectfully disagrees submits that Peterson and Fong focuses on different stages of a software distribution system. For example Peterson focuses on obtaining software requirements, identifying software components that fulfills the requirements, addressing constraints and affinities between said plurality of software components and a respective plurality of configuration options that reflect current best practices with regard to the plurality of software components. Peterson discloses deployment of the software [col. 9, lines 32 — 39] but does not specify the methods used to deploy the software. Although Fong focuses on the method of deploying configurable software [p. 8, paragraph 0055], Fong also generally teaches obtaining software requirements [user enters image capture information (e.g., name, description, and destination for the image) about data processing system; p. 8, paragraph 0055] and identifying software components that fulfills the requirements [p. 1-2, paragraph 0013]. Therefore, the inventions of both

> Peterson and Fong are in the same field of endeavor and it would have been obvious to a person of ordinary skill in the art to combine the references [as to the motivation for combining, see the rejection to claim 1 below].

Final Office Action mailed June 15, 2006, pages 2-3.

Regarding **Peterson's** alleged disclosure of deployment of software, this reference states:

> The customer (organisation) orders the system from a system supplier (in this context the system designer (organisation), but could be a retailer). The customer or supplier organisation delivers the system to the user organisation. The customer and the user often belong to the same organisation, which is one cause of confusion.

Peterson, column 9, lines 33-38.

This excerpt is used by **Peterson** to define the difference between a customer and a user. However, taken at face value, this excerpt discloses that a system is being supplied, not software for loading onto a computer. **Peterson** provides no disclosure whether the system includes software or whether any software is already on the system or supplied separately. Contrary to the interpretation given in the examiner's response, this excerpt does not disclose that this reference is deploying software.

Further, regarding **Fong's** alleged ability to teach obtaining software requirements and identifying software components that fulfills the requirements, this reference states:

> [0013] The present invention provides a comprehensive method and system to facilitate the intelligent deployment of a group of data processing systems with a specific set of software, hardware firmware versions, and parameters under the centralized control of a graphical user interface (GUI). The invention can be implemented in numerous ways, such as by a method, a computer network, or a computer program on electronically-readable media. Three aspects of the invention are described below.
>
> [0055] FIG. 7 illustrates a flow chart of image capture, in accordance with one preferred embodiment of the invention. The method starts in operation 702. In operation 704, the user selects image capture (e.g., from a GUI or menu). Operation 706 is next, where the user selects the reference data processing system. Operation 708 is next, where the user enters image capture information (e.g., name, description, and destination for the image) about data processing system(s). Operation 710 is next, where a test determines if a default image capture, or a customized image capture, is to be made. If the test of operation 710 determines it is a default image capture (i.e., if the user selected the default image capture option), then operation 712 is next, where there is an automatic image

> capture of all hardware configurations and images from the selected reference data processing system. If the test of operation 710 determines it is a customized image capture, then operation 714 is next, where the user selects the customize option. Operation 716 is next, where hardware, base software image, or incremental image capture options are selected. Operation 720 is next, where the image capture status is displayed. Operation 722 is next, where the final report on the image capture is displayed. Operation 724 is next, where the method ends.

Fong, paragraphs 13 and 55.

These excerpts disclose receiving choices from a user that define the image that will be captured. These excerpts do not, however, disclose that *requirements* are received and translated into *software components*. Instead, the user specifies the image to be downloaded. **Fong** is receiving specific choices of elements or groups of elements for copying to a disk image. The choice of specific elements is left to the user, not to the system. Therefore, **Fong** does not teach a computer implemented method of performing the claimed step, as recited in the claim 1.

Thus, one of ordinary skill in the art would not be motivated to combine these two references in the manner suggested by the Examiner. Instead, the references can be combined only through the improper use of hindsight with the benefit of Appellants' disclosure as a template to reach the presently claimed invention.

Therefore, the examiner has failed to state a prima facie obviousness rejection of claims 1-2, 4-47, 49-50. Accordingly, the rejection of claims 1-2, 4-47 and 49-50 under 35 U.S.C. § 103(a) has been overcome.

A.2. Claims 11, 29, 44

Claim 11 is representative of this group. Claim 11 is as follows:

> 11. The computer-implemented method of claim 1, further comprising: generating a difference image that represents differences between the disk image and another existing disk image, whereby the another existing disk image may be updated to match the disk image by applying the difference image to the another existing disk image.

Against this claim, the rejection states:

> 15. As to claim 11, Peterson as modified teaches generating a difference image that represents differences between the disk image and another existing disk image, whereby the another existing disk image may

> be updated to match the disk image by applying the difference image to the another existing disk image [p. 3, paragraph 0036 of Fong].

Final Office Action mailed June 15, 2006, page 6.

The determination of "nonobviousness" is made after establishing the scope and content of prior art, the differences between the prior art and the claims at issue, and the level of ordinary skill in the pertinent art. *Graham v. John Deere*, 383 U.S. 1 (1966). In addition, all limitations of the claimed invention must be considered when determining patentability. *In re Lowry*, 32 F.3d 1579, 1582, 21 U.S.P.Q.2d 1031, 1034 (Fed Cir. 1994). A case of *prima facie* obviousness has not been demonstrated in the rejection of this claim because neither of the references discloses the specific features of this claim.

Claims 11, 29, and 44 are dependent respectively on claims 1, 19, and 34, argued in the rejection above. Claims 11, 29, and 44 distinguish over the references cited for the same reasons as their parent claims. Specifically, considered as a whole, neither **Peterson** nor **Fong** disclose or suggest "*determining ... a plurality of software components that currently exist and that will fulfill the software requirements*"; these references also do not disclose or suggest that the determining is performed "*while addressing constraints and affinities between said plurality of software components*"; and these references further do not disclose or suggest "*determining ... a respective plurality of configuration options that reflect current best practices with regard to said plurality of software components*".

In addition to the features of the invention that are not shown by **Peterson** and **Fong**, there are considerable differences between **Peterson** and **Fong**. Therefore, one of ordinary skill in the art would not seek to combine these references when they are considered as a whole.

Further, the specific features of claim 11 are not met by the art relied on. **Peterson** and **Fong** do not disclose *generating a difference image*, as claimed. The excerpt from **Fong** cited as teaching claimed this feature states:

> [0036] Preferred embodiments of the present invention can deploy multiple partitions and multiple disk drives. The difference between multiple partitions and multiple disk drives is best illustrated by figures. FIG. 2A illustrates partitions on a disk drive X with partitions A, B, and C.

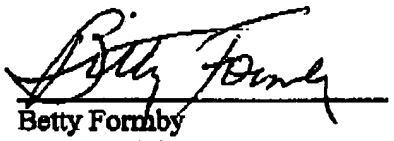
> FIG. 2B illustrates two disk drives X and Y with partitions A and B on drive X, and partitions C and D on drive Y, respectively. FIG. 2C illustrates a partition A shared across multiple disk drives X and Y.

Fong, paragraph 0036.

This excerpt does not address *a difference image that represents differences between the disk image and another existing disk image*. While it is true that the three figures show drawings of disk drives that are different from each other, none of the cited images represent the differences between two other images. Appellants can see no relevance of the disclosed feature in **Fong** to the claimed feature, which is a *difference image*. Moreover, **Patterson** is devoid of disclosure in this regard. Accordingly, the proposed combination when considered as a whole does not teach or suggest all of the features of claim 11. For this reason, the examiner has failed to state a prima facie obviousness rejection of claim 11. Therefore, the rejection of claim 11 and of the other claims in this group should be overturned.

B. CONCLUSION

As shown above, the examiner has failed to state a prima facie obviousness rejection against any of the claims. Therefore, Applicants request that the Board of Patent Appeals and Interferences reverse the rejections. Additionally, Applicants request that the Board direct the examiner to allow the claims.



Betty Formby
Reg. No. 36,536
YEE & ASSOCIATES, P.C.
PO Box 802333
Dallas, TX 75380
(972) 385-8777

CLAIMS APPENDIX

The text of the claims involved in the appeal are:

1. A computer implemented method for creating customized disk images for loading software onto a computer, the method comprising the steps:

receiving software requirements for a given computer system from a plurality of users;

determining (a) a plurality of software components that currently exist and that will fulfill the software requirements while addressing constraints and affinities between said plurality of software components and (b) a respective plurality of configuration options that reflect current best practices with regard to said plurality of software components; and

generating a disk image containing said plurality of software components configured according to said respective plurality of configuration options.

2. The computer-implemented method of claim 1, wherein said determining step applies rules to the software requirements to identify software components that comply with the software requirements.

4. The computer-implemented method of claim 2, wherein the rules include rules mapping a software requirement into a corresponding software component.

5. The computer-implemented method of claim 2, wherein the rules include rules specifying when particular versions of a particular software component are to be utilized.

6. The computer-implemented method of claim 2, wherein the rules include rules specifying installation options regarding a particular software component.

7. The computer-implemented method of claim 2, wherein the rules include rules specifying how to test a particular software component.

8. The computer-implemented method of claim 1, further comprising:

testing the disk image.

9. The computer-implemented method of claim 8, wherein testing the disk image includes verifying that said plurality of software components complies with the software requirements.

10. The computer-implemented method of claim 8, wherein testing the disk image includes verifying that said plurality of software components complies with at least one rule.

11. The computer-implemented method of claim 1, further comprising:

generating a difference image that represents differences between the disk image and another existing disk image, whereby the another existing disk image may be updated to match the disk image by applying the difference image to the another existing disk image.

12. The computer-implemented method of claim 1, wherein the software requirements are received through a network that includes the Internet.

13. The computer-implemented method of claim 1, wherein the software requirements can be received in terms of customer needs rather than specific software components.

14. The computer-implemented method of claim 1, wherein the requirements are represented in a structured format.

15. The computer-implemented method of claim 14, wherein the structured format is Extensible Markup Language (XML).

16. A computer-implemented method for creating a customized disk image for loading software onto a computer, the method comprising the computer-implemented steps:

parsing a plurality of inputs regarding a desired computer system to extract specifications regarding software;

evaluating a plurality of rules with respect to the plurality of inputs to derive a set of software components conforming to the specifications, said set of software components being chosen from existing software components;

evaluating a second plurality of rules with respect to the plurality of inputs to derive a set of configuration options conforming to at least the specifications;

storing each software component from the set of software components on a storage device;

configuring each software component stored on the storage device in accordance to the set of configuration options; and

generating a disk image from contents of the storage device.

17. The computer-implemented method of claim 16, wherein the plurality of inputs are requests from hypertext browsers.

18. The computer-implemented method of claim 16, wherein the plurality of inputs are XML documents.

19. A computer program product stored in a computer-readable medium and comprising functional descriptive data that, when executed by a computer, enables the computer to create customized disk images for loading software onto a computer, including the steps:

receiving software requirements for a given computer system from a plurality of users;

determining (a) a plurality of software components that currently exist and will fulfill the software requirements while addressing constraints and affinities between said plurality of software components and (b) a respective plurality of configuration options that reflect current best practices with regard to said plurality of software components; and

generating a disk image containing said plurality of software components configured according to said respective plurality of configuration options.

20. The computer program product of claim 19, wherein said determining step applies rules to the software requirements to identify software component that comply with the software requirements.

21. The computer program product of claim 20, wherein the rules are stored in a database.

22. The computer program product of claim 21, wherein the rules include rules mapping a software requirement into a corresponding software component.

23. The computer program product of claim 21, wherein the rules include rules specifying when particular versions of a particular software component are to be utilized.

24. The computer program product of claim 21, wherein the rules include rules specifying installation options regarding a particular software component.

25. The computer program product of claim 21, wherein the rules include rules specifying how to test a particular software component.

26. The computer program product of claim 19, comprising additional functional descriptive data that, when executed by the computer, enables the computer to perform additional acts including:

testing the disk image.

27. The computer program product of claim 26, wherein testing the disk image includes verifying that said plurality of software components complies with the software requirements.

28. The computer program product of claim 26, wherein testing the disk image includes verifying that said plurality of software components complies with at least one rule.

29. (Original) The computer program product of claim 19, comprising additional functional descriptive data that, when executed by the computer, enables the computer to perform additional acts including:

generating a difference image that represents differences between the disk image and another existing disk image, whereby the another existing disk image may be updated to match the disk image by applying the difference image to the another existing disk image.

30. The computer program product of claim 19, wherein the software requirements are received through a network that includes the Internet.

31. The computer program product of claim 19, wherein the software requirements can be received in terms of customer needs rather than specific software components.

32. The computer program product of claim 19, wherein the requirements are represented in a structured format.

33. The computer program product of claim 32, wherein the structured format is Extensible Markup Language (XML).

34. A data processing system capable of creating customized disk images for loading software onto a computer, said data processing system comprising:

instructions for receiving software requirements for a given system from a plurality of users;

instructions for determining (a) a plurality of software components that currently exist and that will fulfill the software requirements while addressing constraints and affinities between said plurality of software components and (b) a respective plurality of configuration options that reflect current best practices with regard to said plurality of software components; and

instructions for generating a disk image containing said plurality of software components configured according to said respective plurality of configuration options.

35. The data processing system of claim 34, wherein said instructions for determining applies rules to the software requirements to identify software component that comply with the software requirements.

36. The data processing system of claim 35, wherein the rules are stored in a database.

37. The data processing system of claim 36, wherein the rules include rules mapping a software requirement into a corresponding software component.

38. The data processing system of claim 36, wherein the rules include rules specifying when particular versions of a particular software component are to be utilized.

39. The data processing system of claim 36, wherein the rules include rules specifying installation options regarding a particular software component.

40. The data processing system of claim 36, wherein the rules include rules specifying how to test a particular software component.

41. The data processing system of claim 34, further comprising:

means for testing the disk image.

42. The data processing system of claim 41, wherein testing the disk image includes verifying that said plurality of software components complies with the software requirements.

43. The data processing system of claim 41, wherein testing the disk image includes verifying that item said plurality of software complies with at least one rule.

44. The data processing system of claim 34, further comprising:

means for generating a difference image that represents differences between the disk image and another existing disk image, whereby the another existing disk image may be updated to match the disk image by applying the difference image to the another existing disk image.

45. The data processing system of claim 34, wherein the software requirements are received through a network that includes the Internet.

46. The data processing system of claim 34, wherein the software requirements can be received in terms of customer needs rather than specific software components.

47. The data processing system of claim 34, wherein the requirements are represented in a structured format, such as Extensible Markup Language (XML).

49. The computer implemented method of claim 1, further comprising storing said disk image on a computer-readable media and distributing said computer-readable media to a client.

50. The data processing system of claim 34, further comprising instructions for storing said disk image on a computer-readable media, wherein said computer-readable media can be distributed to a client.

EVIDENCE APPENDIX

There is no evidence to be presented.

RELATED PROCEEDINGS APPENDIX

There are no related proceedings.